Introducing novice programmers to functions and recursion using computer games

Law, Bobby

Published in:
ECGBL 2018 12th European Conference on Game-Based Learning

Publication date:
2018

Document Version
Author accepted manuscript

Introducing Novice Programmers to Functions and Recursion using Computer Games.

Robert Law
Glasgow Caledonian University, Glasgow, Scotland
robert.law@gcu.ac.uk

Abstract: Having got to grips with the basic building blocks of programming, novice programmers must progress to more challenging concepts such as functions and recursion. To get the most from their programming journey novice programmers must attempt to bridge the gap between the solid foundation they have built and the more challenging programming concepts that will help expand their programming armoury. Functions and recursion can present a stumbling block for the novice programmer as they can be difficult to visualise, especially, recursion. The novice programmer can find it difficult to understand the flow of control of a program and this can be compounded by having to understand the backward flow of control of recursion. This unfamiliarity with recursive activities, like Alice, can lead the novice programmer down a rabbit hole of confusion from which they find it difficult to escape, shattering their confidence.
The stumbling block for novice programmers is building a mental model of the program stack and subsequently identifying the backward flow of control from the last invocation of the recursive method.
Novice programmers can find themselves caught in a web of code that helps little with their ability to understand this powerful programming technique. Hence, the ability to visualise such a technique could open the door to better understanding for the novice programmer. There are many computer games available which are designed to teach programming concepts using a visual interface, requiring the programmer to apply problem solving and computational thinking to derive a solution to the puzzle posed. Such programs offer a way to visualise abstract concepts such as recursion and offer a solution mechanism that does not require the novice programmer to implement their solution steps using a programming language. Offering a program language agnostic approach, the novice programmer, can build an understanding of the concept of recursion such that they should be able to develop an algorithm that could be programmed using any programming language.
This paper will discuss an ongoing attempt to introduce functions and recursion to second year Game Design students undertaking a C++ game development module.

Keywords: Programming, Problem Solving, Recursion, Functions, Methods, Procedures

1. Introduction

For the novice programmer, taking the next step to understand higher level programming concepts such as recursion and functions/methods can be a daunting prospect. Possibly, still consolidating their understanding of the fundamental building blocks of programming, if this foundation is not capable of taking the weight of higher level concepts the student may find it difficult to grasp the complexities of such concepts.

McCauley et al. (2015) suggest that educators have identified recursion as a challenging topic for novice programmers to comprehend. It is vital, therefore, that novice programmers are able to understand and comprehend the underlying concept of recursion without the burden of a specific programming language.

As suggested by Law (2017) presenting the student with a language agnostic approach to teaching programming concepts and applying a problem solving approach leaves the door open for the use of games specifically designed for teaching programming concepts.

Rajaravivarma (2005) espouses the use of computer games specifically designed to teach programming as they offer the student the ability to problem solve and learn programming concepts in a visual manner.

The remainder of this paper is organized as follows: Section 2 gives information about pedagogical issues related to recursion and functions/methods, Section 3 introduces the online programming game used with the students and the subsequent task undertaken with the students. Section 4 discusses the results garnered from surveying the students with regard to the tasks introduced in Section 3. Section 5 offers a summary of the student experience of using the selected programming game, and concludes the paper giving proposals for future work.

2. Literature Review

Lee et al. (2014) declare that recursion is a "fundamental concept" with Ginat & Shifroni (1999) suggesting that it is "powerful" and necessary "computational problem solving tool". Both conclude that it is a concept that novice programmers can find taxing to understand. This claim is substantiated by McCauley et al. (2015) who point out that a number of educators and researchers "consider recursive programming challenging for novices.".

Dann et al. (2001) note an interesting paradox which suggests that building a solid understanding of recursion early within a course can be a benefit as this can make understanding concepts presented in later modules easier to understand, unfortunately, novice programmers wrestle with the concept of recursion. It would, therefore, seem important to introduce novice programmers to the concept of recursion early, hoping that they understand the concept. Ginat & Shifroni (1999) allude to the point of introduction for the concept of recursion as late within a course which contradicts the early introduction advocated by Dann et al. (2001). The basic form of recursion is normally taught during year 1 and/or 2 of a typical programming course (Gunion et al. 2009).

Students misconceptions of recursion are compounded by the fact that they build inaccurate models of the recursive process confusing and associating recursion with an iterative process (Lee et al. 2014). Further, Lee et al. (2014) also suggest that novice programmers are unable to fully comprehend "the use of the stack for backtracking".

Wu et al. (1998) present five conceptual models which have been used to teach recursion. These models are Russian Dolls, Process Tracing, Stack Simulation, Mathematical Induction and Structural Template. Russian Dolls, Process Tracing and Stack Simulation can be classified as concrete models with Mathematical Induction and Structural Template classified as abstract models (Wu et al. 1998). Each of these models offers a different approach which may suit different learners. The Russian doll exhibits recursion by dismantling the doll to produce smaller and smaller exact copies within each other until the recursive process stops when the last doll doesn't contain another. Process Tracing attempts to teach the novice programmer recursion by showing the "process generated by recursive functions" (Wu et al. 1998). Stack Simulation uses the computers architecture, in particular, the system stack to show recursion in action. Mathematical induction is firmly ground in mathematical proofs. Structural Template attempts to show recursion by the use of "samples of recursive programs and describes the base cases and recursive cases." (Wu et al. 1998).

Tessler et al. (2013) suggest that recursion is introduced to students initially by playing a video game, in this case, Cargo-bot. Their approach is based on the idea of "contextualized learning", a technique that proposes that novice programmers will understand new concepts better when they can relate these to formerly understood concepts (Tessler et al. 2013). Interestingly, Tessler et al. (2013), mention that there are "few natural instances of recursion" that would be obvious to students in their everyday live but, the number of computer games that explicitly need the "player to think recursively" is on the increase.

Gunion et al. (2009) indicate that recursion is a an efficient mechanism for implementing sorting and searching solutions and is a valuable problem solving technique, citing visual representations of recursion can be seen in fractals, Sierpinski's Carpet and the puzzle game Towers of Hanoi. Both Gunion et al. (2009) and Butgereit (2016) suggest that examples of recursion can be found in nature quoting "the structure of petals in flowers" and the "organization of planets and moons in our solar system". Butgereit (2016) also suggests that recursion can "model the spread of epidemics".

Gouws et al. (2013) created their own Computational Thinking Framework (CTF) which they used to assess the suitability of LightBot (version 1). This framework identified six key areas; Processes and Transformations, Models and Abstractions, Patterns and Algorithms, Tools and Resources, Inference and Logic, and finally, Evaluations and Improvements. This framework and a four point likert scale, which Gouws et al. (2013) acknowledge can be subject, were used to evaluate LightBot (version 1) obtaining an overall Computational Thinking Score of 74%. Thus, the authors assessment that LightBot (version 1) was a "useful game for practicing computational thinking".

3. Overview of Pilot Study

The literature review would suggest that there is plausibility in the hypothesis that games tailored for teaching programming can offer students a credible platform for building on the programming concepts sequence, selection and iteration and introducing higher order concepts such as procedures/functions and recursion. Ideally, the preference would have been to use Kazimoglu et al. (2012) "serious game", the web based Program Your Robot, as detailed in Law (2017), however, this web based game is no longer available. Thus, a suitable replacement game was sought.

3.1 Selecting an Appropriate Game

To select an appropriate replacement game Vahldick et al. (2014) list of 40 games, based on the Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science (ACM & IEEE 2013), was revisited. The criteria for selecting a replacement game was again based on Law (2017): web-based, matching ACM & IEEE Fundamental Programming Concepts 1.6 & 1.7 (Functions and Recursion) (ACM & IEEE 2013) and primarily free. A number of games were revisited for suitability; BotLogic, Lightbot 2.0 and Gidget. The decision was taken to use Light-Bot 2.0, a web based game, covering ACM & IEEE Fundamental Programming Concepts 1.6 & 1.7.

In relation to problem solving Cazzola & Olivares (2016) highlight an interesting dichotomy; if the teaching tool used is “too complex” students will reduce their time spent on problem solving in order to focus more on the tool being used. This is an unwanted distraction and any game used should be easy to use, letting the student concentrate on solving the problem and not how to use the game.

Opting for Light-Bot2 offers a web based game with a similar look and feel to Program Your Robot as used by Law (2016; 2017). Lightbot 2.0 is a flash based game available from a number of sources; the version used during this work is available at http://coweb.cc.gatech.edu/ice-gt/1835. This game introduces and reinforces the concepts of procedures/functions and recursions as well as reiterating the programming fundamentals of sequence, selection and iteration. Lightbot 2.0 is a cross-platform game developed by Danny Yaroslavski becoming popular with all age ranges and redeveloped in 2013 to be education friendly.

Combéfis et al. (2016) state that, in an educational context, feedback is imperative. Lightbot offers both visual feedback and debugging as the students can envision their programs execution (Combéfis et al. 2016).

As stated by Law (2016; 2017) this type of environment offers the learner a programming language agnostic approach stressing the need for problem solving while reinforcing the programming concepts of sequence, selection and iteration.

In a similar vein to Program Your Robot, Lightbot introduces the learner to the programming concepts of sequence, selection, iteration, functions/methods and recursion. The student must build an algorithm using a combination of commands to enable the Robot to complete its designated task and thus progress to the next level (Law 2016).

Figure 1 shows the menu options available to the student. The student can start with any of “Basics”, “Recursion”, “Conditionals” or “Expert” levels. “Basics”, “Recursion” and “Conditionals” each have one level open for play and more levels are unlocked as the student completes each previous level. Unlike Program your Robot which offers a gradual introduction to the programming concepts, students may be tempted to skip the fundamental concepts and instead dive into a more difficult topic.

Combéfis et al. (2016) note that the purpose of Lightbot is "to teach programming concepts" rather than "a platform to learn how to code". This, they suggest, fits well with the concept of "Algorithmic Thinking". Playing games, such as Lightbot, provide the student with a puzzle based challenge which, in turn, should help promote "Algorithmic Thinking" for the student (Combéfis et al. 2016).

A small selection of commands is available for the student to use when building their proposed solution algorithm. The commands consist of movement commands to manoeuvre the robot and commands to call the two available functions/methods. The exposure to functions/methods offers the student the ability to create

reusable command templates which helps reinforce the need to decompose a problem into manageable chunks as postulated by Winslow (1996).

Figure 2 shows the programming learning environment for the game. On the right hand side of the user interface there is a main method, the prime control mechanism for controlling the robot, and two functions/methods. On the top right hand side of the user interface are the commands which can be used to control the robot. The user interface permits drag and drop of any of the commands to fill the empty slots within the main method or the two functions/methods. A Run button is located at the bottom right of the user interface and is used to execute the solution algorithm. The user interface is reasonably clear and self-explanatory. During the early levels, the game gives the student a number of prompts to help with build a suitable solution.

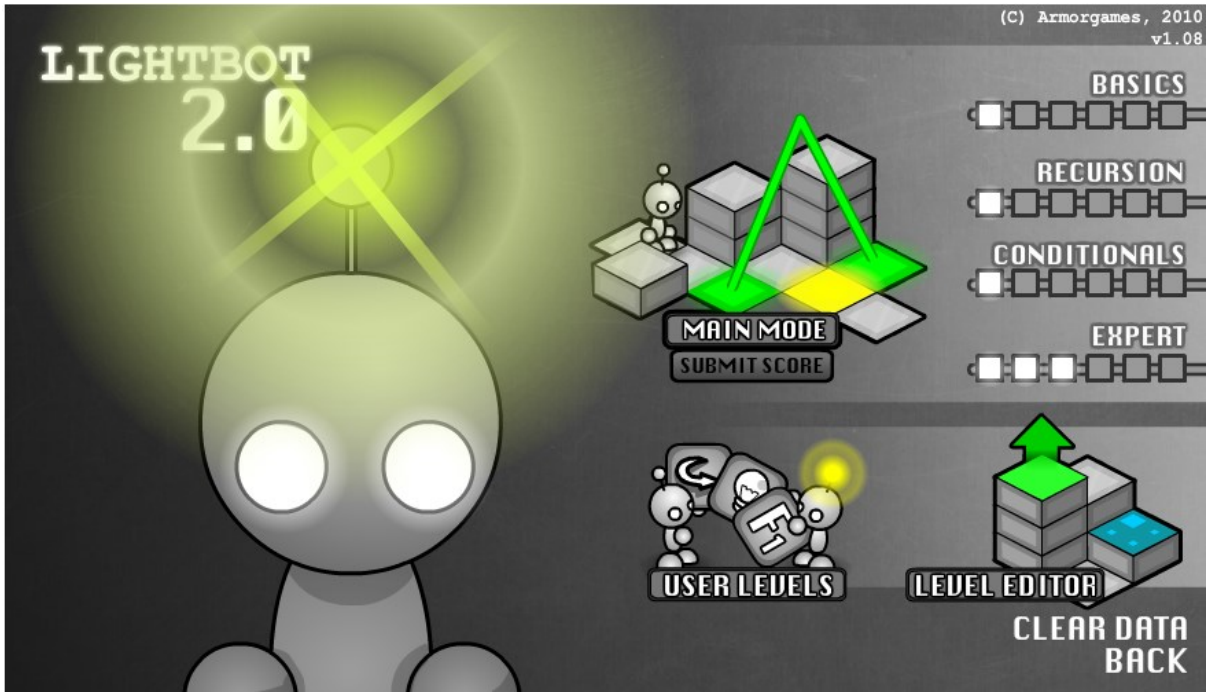

Figure 1 Lightbot 2.0

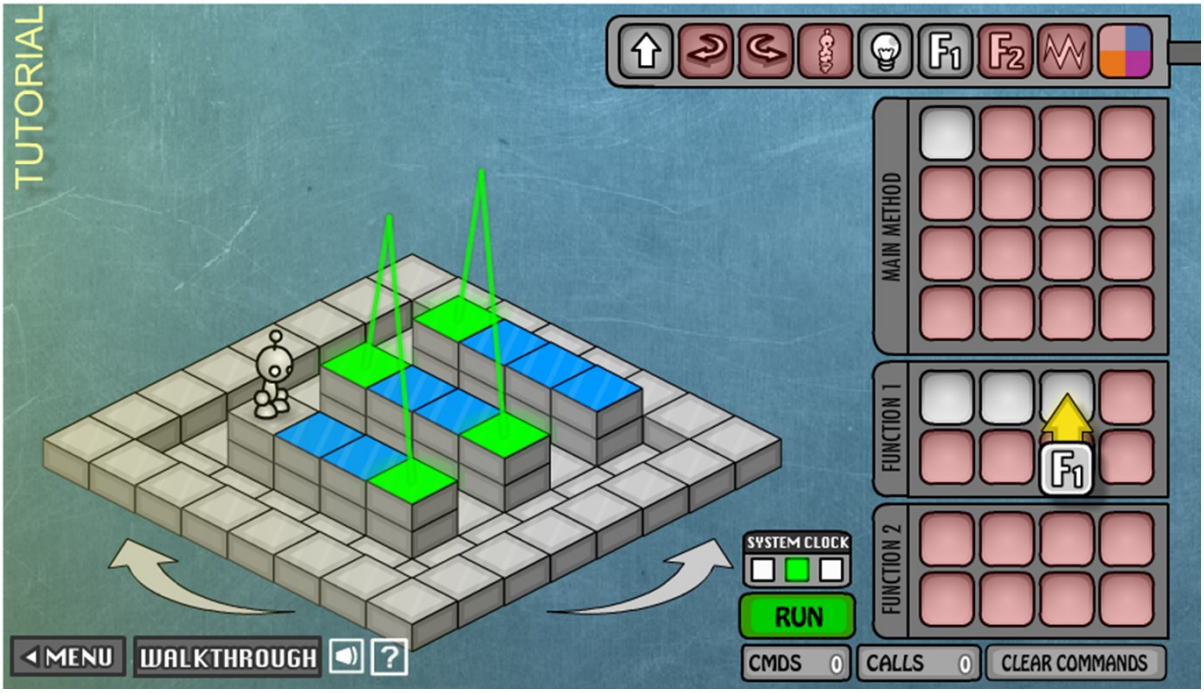

Figure 2 Lightbot 2.0 Recursion

As with Program Your Robot, each level offers the student a different number of free slots for creating an effective solution algorithm, thus, the student sees the need to use functions/methods to develop reusable algorithms Law (2016; 2017).

3.2 Student Task

The students undertaking the pilot study consist of two cohorts, both second year undergraduates, the larger of the two are part of the Game Design (GD) programme and the smaller of the groups are part of the Game Software Development (GSD) programme totalling 45 students. The students had had one lecture and one tutorial each for both functions/methods and recursion.

The task presented to the students was quite open ended; they were given the web address for Lightbot 2.0 and simply asked to play the game and note how they solved the tasks the game presented to them. On completion of the game the students were asked to complete an online questionnaire, hosted using Google Forms. The students did this exercise as part of a one-hour lab based session.

4. Pilot Study Results

This section will present the findings from the survey, of which 34 of the 45 students enrolled on the module completed, giving a yield of 76%. All responses to the survey were anonymous.

The first question on the survey was "What do you think the purpose of the game was?" Although, the students had previously had a formal lecture and tutorial on functions/methods and recursion, they had not been given any information about the game itself. A sample of the responses to this question are given below:

"To show off some main programming aspects (such as recursion) in a more understandable way."
"To help the user understand the concept and use of loops and functions in programming."
"teaching programming concepts in an enjoyable way"
"Teaching and improve logic programming skills"
"the purpose was to plan ahead like pseudo code"
"to tease the brain and get it problem solving"
"To understand how the commands that you put into your methods affect the overall output of the game, which will improve one's understanding of programming"

As can be seen from the sample above, the majority of students understood that the purpose of the game was to teach programming but few were able to hone in on the teaching of recursion and functions/methods. This was disappointing but may be due to the open nature of the question.

Students were subsequently asked if they found the game enjoyable to play. The students had to select a value on a scale of 1 (Not Very) to 5 (Very). Figure 3 indicates the spread of responses.

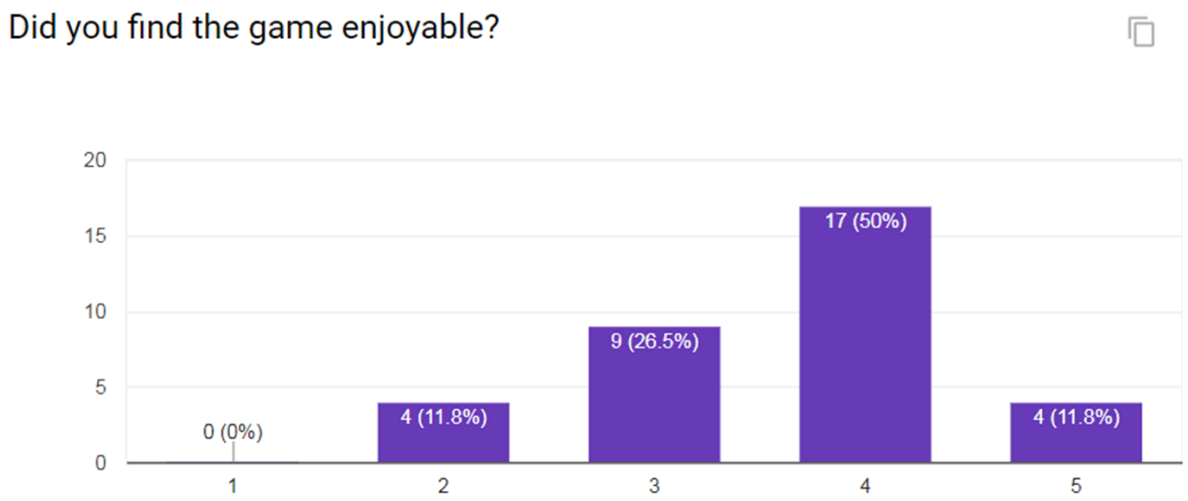

Figure 3 Response to the question "Did you find the game enjoyable?"

The majority of respondents found the game to be enjoyable but four students gave a rating of 2, which would indicate that they did not enjoy the game as much as was expected. This was a departure from the findings in the previous study, Law (2017), undertaken with the cohort in the previous academic year. This may be due to the fact that the game previously used, Program Your Robot, had been specifically designed to teach programming and the replacement game, Lightbot 2.0, although education friendly, has not been designed to the same level.

As with the previous study, Law (2017), it was important to determine if the game was easy to play. Again, the students had to select a value on a scale of 1 (Not Very) to 5 (Very). The graph displayed in figure 4 displpays the results.

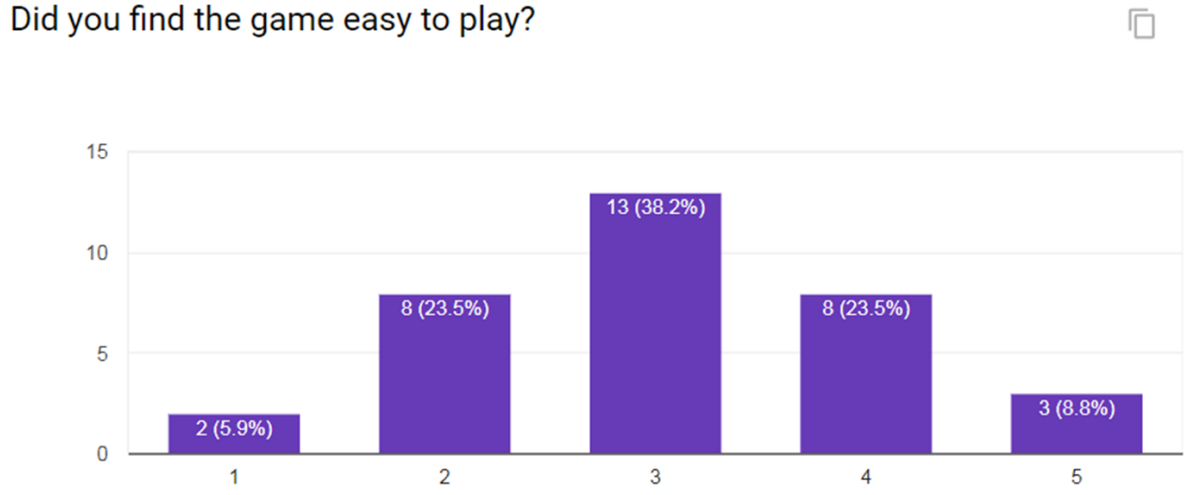

Figure 4 Response to the question "Did you find the game easy to play?"

It was surprising and slightly disappointing to see that almost 30% of respondents did not find the game easy to play. This is a significant increase from the previous work using Program your Robot Law (2017). Video walkthroughs for each level are available, so, it could be surmised that the students who did not find the game easy to play did not watch the videos. It could also be, as suggested above, that Program Your Robot has been better designed.

Students were asked if they completed the game. Students were required to respond to this question with a simple yes or no. The results are illustrated in figure 5.

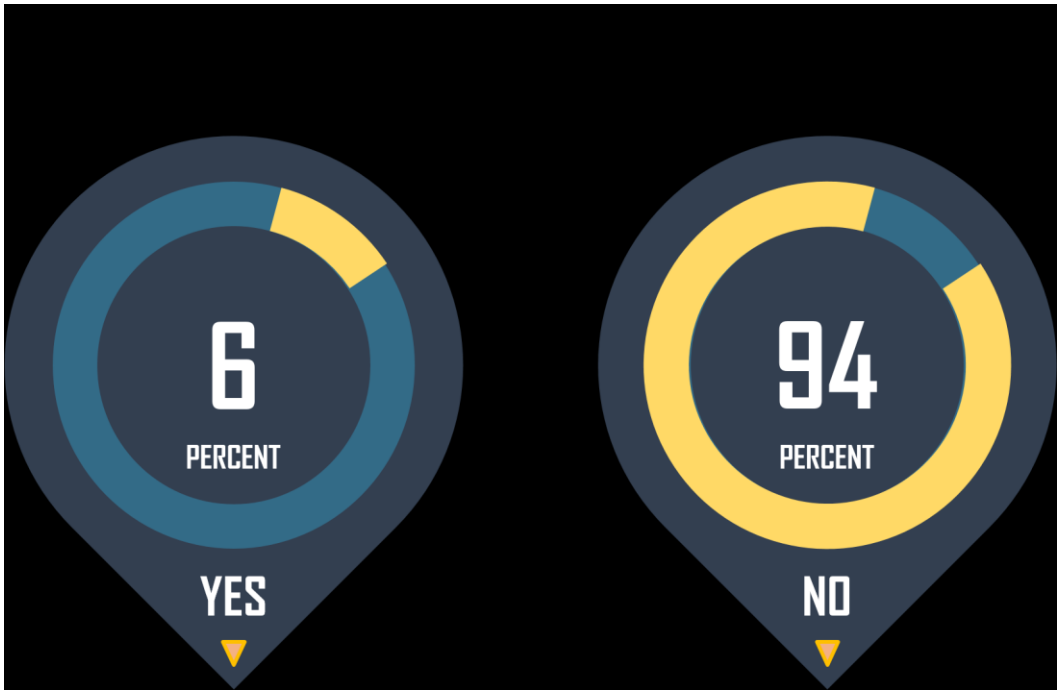

Figure 5 Response to the question "Did you complete the game?"

Staggeringly 94% of the respondents did not manage to complete the game. This was absolutely unexpected. It was a complete reversal of the result in the previous study Law (2017). One possible reason for this is the nature of Lightbot in comparison to Program Your Robot. The former allows the student to "dip in" to the game at various points and the latter has a structured approach that builds competence in each programming concept prior to moving on to the next level.

To try to elicit from the students the programming concepts they perceived the game to be purveying the students were asked to select from a list of programming concepts. The results are displayed in figure 6.

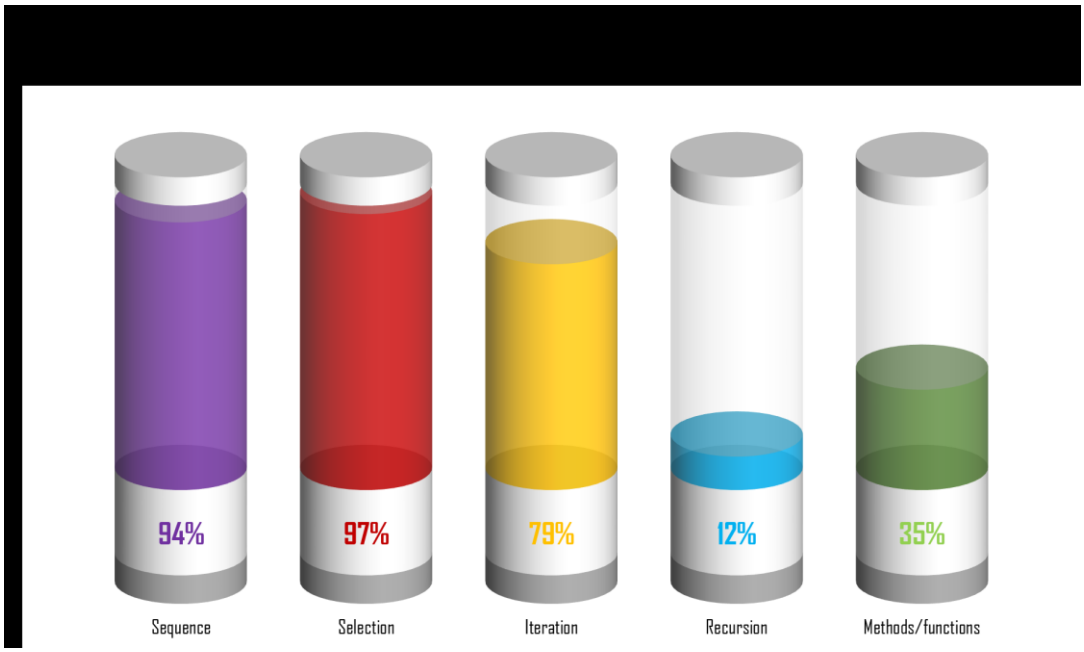

Figure 6 Response to the question "Select the programming concepts you think the game is teaching?"

From the perspective of the fundamental programming concepts of sequence, selection and iteration Lightbot appeared to convey these concepts to the students. However, for recursion and methods/functions, which

were the main focus of the study, surprisingly few students identified those concepts, which is very surprising considering how obvious they appear in both figure 1 and figure 2.

Having played Lightbot, the students were asked to choose from four phrase which one they thought best described the concept of recursion. The phrases are presented below:

A. A function that calls a separate function.
B. A function that executes a sequence of code.
C. A function that calls itself.
D. A function that does nothing.

The hope was that students would identify phrase C as the best description of recursion. The graph in figure 7 indicates the students' responses.

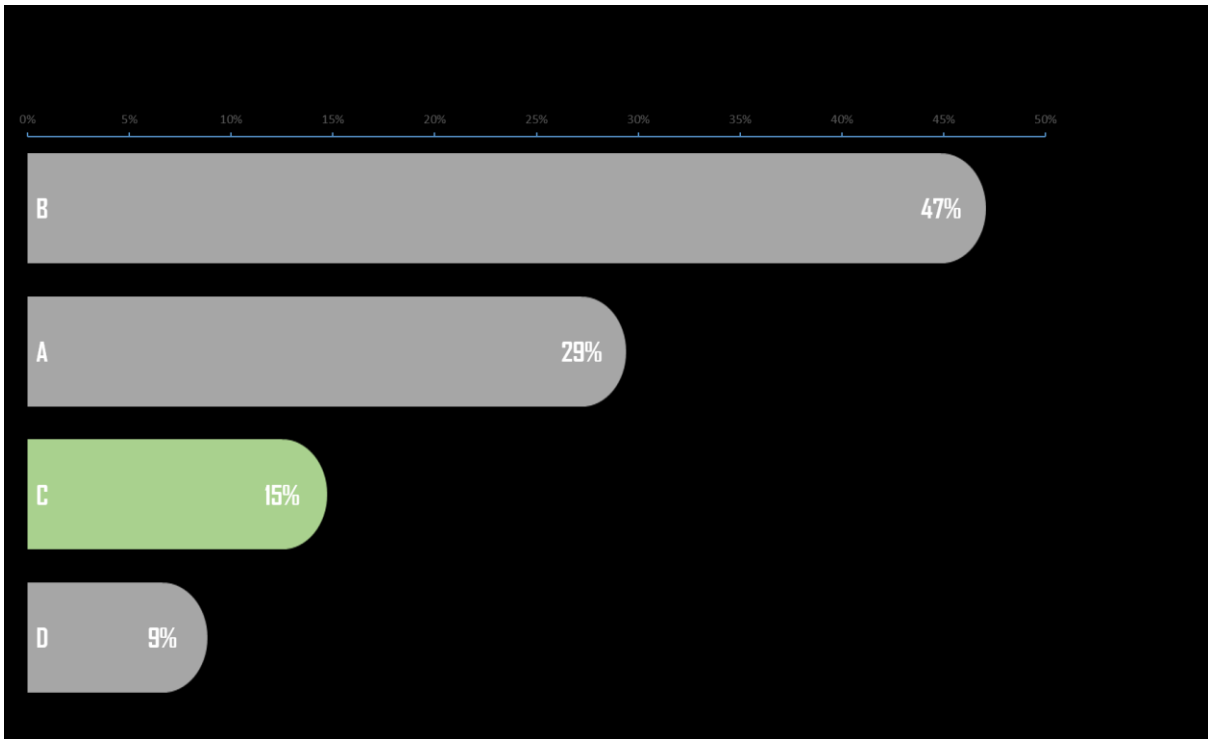

Figure 7 Response to the question "Select the phrase that best describes recursion."

Again, the results proved disappointing, with only 15% of students identifying option C as the best description. This confirms the findings of the literature review, that recursion proves to be a difficult concept for students to grasp. It also raises questions about the effectiveness of the game to convey the concept of recursion in a meaningful manner to the student.

Having played Lightbot, the students were asked to choose from four phrases which one they thought best described the concept of functions/methods. The phrases are presented below:

A. A block of code that does nothing.
B. A block of code that executes a number of related tasks.
C. A block of code that can only be used once.
D. A block of code that executes a specific task and can be reused.

The hope was that students would identify phrase D as the best description of recursion. The graph in figure 8 indicates the students' responses.

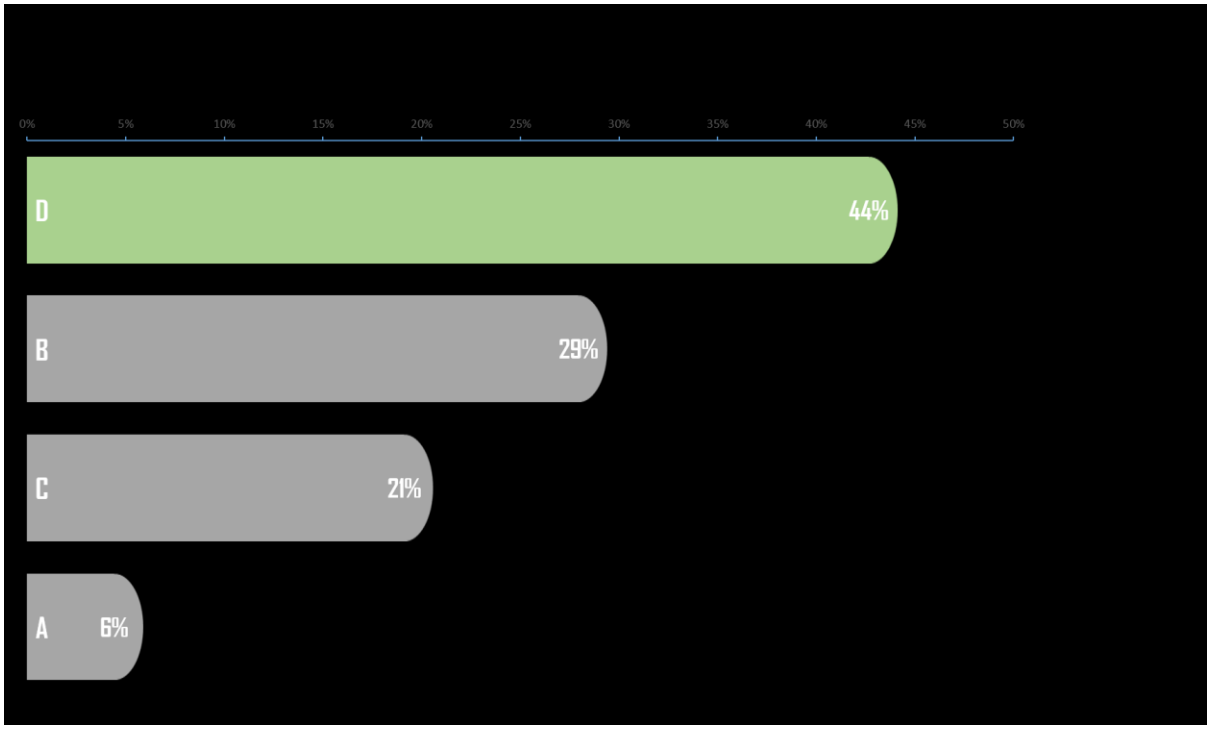

Figure 8 Response to the question "Select the phrase that best describes a procedure."

Although, still disappointing, it was slightly more heartening that 44% of the students did identify the most appropriate phrase. However, by the same token, it is disappointing that 56%, the majority, albeit slim, did not identify a suitable phrase to describe the concept of a function/method.

5. Conclusion

Comparing this study with the study undertaken during the previous academic year Law (2017) it is fair to say that the findings are disappointing, but they do tally with the evidence provided in the literature review, that students find the concept of recursion, in particular, and the concept of functions/methods difficult to understand.

Figure 6 would suggest that Lightbot was capable of delivering the fundamental building blocks of programming but, for this cohort, did not convey the concepts of recursion and functions/methods with the same level of clarity.

The study is a simple snapshot of a particular subsection of one cohort and as such the analysis of the results are subjective and open to interpretation. However, the results would suggest that the students did not, after playing this game, have a better understanding of the concepts of recursion and functions/methods. Therefore, it would be fair to say, that the initial hypothesis was not really achieved. This was very disappointing.

Interestingly, Gouws et al. (2013), observe that "novice" programmers may not appreciate the similarities between LighthBot and programming. To this end, further work is required to ascertain the students ability to "transfer general computational thinking skills to programming" (Gouws et al. 2013).

Honing in on Gouws et al. (2013) Computational Thinking Framework, with particular focus on Patterns and Algorithms, they make explicit reference to recursion and functions, however, through their analysis they conclude that "Light-Bot has great potential for reinforcing the use of patterns and algorithms, we feel that in practice it fails to address these on a meaningful level." A caveat to this is the fact that, by their own admission, LightBot 2.0 has went someway to address "some of these shortcomings".

The study and its delivery will be refocused for the next academic year. The refocus will be influenced by Lee et al. (2014) work such that a better set of learning outcomes will be devised with specific regard to recursion,

these will be mapped onto the puzzles offered by LightBot and the Lecture and tutorial covering recursion will use examples from LightBot. Also, the focus of the online questionnaire will be tightened to match the learning outcomes.

Other possible games will be evaluated using Vahldick et al. (2014) list of games and criteria and also Gouws et al. (2013) Computational Thinking Framework (CTF).

References

ACM & IEEE, 2013. Computer Science Curricula 2013.

Butgereit, L., 2016. Teaching recursion through games, songs, stories, directories and hacking. In *Advances in Computing and Communication Engineering (ICACCE), 2016 International Conference on*. pp. 401–407.

Cazzola, W. & Olivares, D.M., 2016. Gradually learning programming supported by a growable programming language. *IEEE Transactions on Emerging Topics in Computing*, 4(3), pp.404–415.

Combéfis, S., BERESNEVI\vCIUS, G. & Dagien\.e, V., 2016. Learning programming through games and contests: overview, characterisation and discussion. *Olympiads in Informatics*, 10(1), pp.39–60.

Dann, W., Cooper, S. & Pausch, R., 2001. Using visualization to teach novices recursion. *ACM SIGCSE Bulletin*, 33(3), pp.109–112.

Ginat, D. & Shifroni, E., 1999. Teaching recursion in a procedural environment—How much should we emphasize the computing model? In *ACM SIGCSE Bulletin*. pp. 127–131.

Gouws, L.A., Bradshaw, K. & Wentworth, P., 2013. Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. pp. 10–15.

Gunion, K., Milford, T. & Stege, U., 2009. Curing recursion aversion. In *ACM SIGCSE Bulletin*. pp. 124–128.

Kazimoglu, C. et al., 2012. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9, pp.522–531.

Law, B., 2016. Puzzle Games: A Metaphor for Computational Thinking. In *10th European Conference on Games Based Learning: ECGBL 2016*. p. 344.

Law, R., 2017. Teaching Programming Using Computer Games: A Program Language Agnostic Approach. In *ECGBL 2017 11th European Conference on Game-Based Learning*. p. 368.

Lee, E. et al., 2014. A structured approach to teaching recursion using cargo-bot. In *Proceedings of the tenth annual conference on International computing education research*. pp. 59–66.

McCauley, R. et al., 2015. Teaching and learning recursive programming: a review of the research literature. *Computer Science Education*, 25(1), pp.37–66.

Rajaravivarma, R., 2005. A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, 37(4), pp.98–102.

Tessler, J., Beth, B. & Lin, C., 2013. Using cargo-bot to provide contextualized learning of recursion. In *Proceedings of the ninth annual international ACM conference on International computing education research*. pp. 161–168.

Vahldick, A., Mendes, A.J. & Marcelino, M.J., 2014. A review of games designed to improve introductory computer programming competencies. In *Frontiers in Education Conference (FIE), 2014 IEEE*. pp. 1–7.

Winslow, L.E., 1996. Programming pedagogy—a psychological overview. *ACM Sigcse Bulletin*, 28(3), pp.17–22.

Wu, C.-C., Dale, N.B. & Bethel, L.J., 1998. Conceptual models and cognitive learning styles in teaching recursion. In *ACM SIGCSE Bulletin*. pp. 292–296.